CSE222 / BİL505
Data Structures and Algorithms
Assignment #3
Due Date: 30/03/2025, 23:59

In this assignment, you will use the code you wrote for the previous assignment and add additional features for logging and output handling. Also, you are expected to document the software you wrote properly, by adding comments and generating JavaDoc documentation.

Input Handling

The input commands must be taken only from the terminal. The commands received should not be processed immediately but stored in a List data structure. After the exit command is received, the program should execute all commands received in the same order they were entered. You need to strictly follow the command format in the previous assignment, any failure in input handling will fail the test scenarios.

Input rules:

1. The configuration file must be taken as a command line argument, the program should not be launched without it.
2. One command should be read at a time, failing to do that might fail the tests.
3. Input format should strictly follow the description in the previous assignment. Tests will be done using scripts. Be careful with whitespaces and upper/lower case characters.

Output Handling

After receiving the exit command, the program should start executing commands and printing the output to the terminal. Only a single newline character should exist between the outputs of two commands. The commands' outputs should strictly follow these formats:

- turnON/turnOFF: "<devName>: Turning ON/OFF."
- addDev: "Device added."
- rmDev: "Device removed."
- readSensor: As it was in the previous homework, but the values should be constants instead of being randomly generated:
    - Temperature: 24.00
    - Acceleration: 1.00
    - Rotation: 0.50
- setMotorSpeed: "<devName>: Setting speed to <Speed>."
- readWireless: "<devName>: Received \"Some Data\"."
- writeWireless: "<devName>: Sending \"<input string>\"."
- printDisplay: "<devName>: Printing \"<input string>\"."

- Exit: should output the string “Exiting ...”

Output rules:

1. The output must exactly match the expected output, even in lower/upper cases and whitespaces. Tests will be done using scripts.
2. Nothing other than the expected output of commands should be printed. (New lines, helper text, etc.)
3. All outputs for successful operations must be written to stdout.
4. Any minor deviation will result in failing the test.

Port Handling

In the previous assignment, object instances of the Protocol class were called ports, no need to get confused about it. In this assignment, ports will not print anything on the terminal but will hold a list of requests that were received during execution. Each protocol instance should create a log file with the name: <protocolName>_<portID>.log. The log file should contain the requests issued to the port in descending order, start with the last and finish with the first. Also, the first record should be “Port Opened.”. Since there are only two operations, a record can either be “Reading.” Or “Writing \”<Input>\”.”. Logs will be written to a path given as a command line argument.

Error Handling

All error messages should be written to stderr. You are free to choose the format that suits you. The more informative the errors are, the better it is to debug.

Complexity

All operations should have time complexity of O(1). Choose the appropriate data structures and operations accordingly.

JavaDoc

You will have to generate the documentation of your code using JavaDoc tool. To make the documentation meaningful, you need to add comments and explain each class and method in your code. Documentation with no explanations is not useful.

Clarifications On the Previous Assignment

All details and restrictions in the previous assignment apply, unless specified otherwise. Here are some additional points:

1. The following methods must only be implemented in classes at the bottom of the hierarchy: turnON, turnoff, getTemp, getAccel, getRot, printData, sendData, recvData. setMotorSpeed.

Rules and Restrictions

Abide by those rules. Any violations will be heavily penalized. The rules are:

- No for-each loops are allowed.
- You must use ArrayList, LinkedList, Queue, and Stack data structures.
- Only use iterators for iterating through ArrayList and LinkedList.
- Allowed libraries:
    - java.io.File
    - java.util.ArrayList
    - java.util.Arrays
    - java.util.Scanner
    - java.util.Iterator
    - java.util.LinkedList
    - java.util.List
    - java.util.Queue
    - Whatever exceptions you are monitoring.
- No OOP violations are allowed.
- All restrictions from the previous homework apply, unless specified otherwise.

General Information About the Homework:

- Cheating is not permitted. The students who cheat will receive NA from the course.
- In case of using AI tools for code generation, please specify it, otherwise, claiming ownership of code not your own will be penalized.
- You will be asked to attend a demo session to be graded. If you do not attend a demo or cannot give proper answers to the questions you were asked during the demo, you will get zero.
- Late submissions will not be allowed. The due date will not be postponed.
- Tests will be performed using OpenJDK 11.
- Use a Linux environment. The test will be automated using scripts. You are responsible for any deviations in input/output format, makefile commands, command line arguments, or directory structure.
- The assignment to be returned must follow this format:
    - A zip archive.
    - The assignment's name should be: <name>_<surname>_<studentNo>.zip.
    - After extraction, makefile commands will be executed from the root directory of the assignment. (Be sure that your files are not in an additional directory inside the archive)
    - Don not put any Turkish characters in the file name.
    - A compatible makefile will be demonstrated in the PS.
- Any questions sent after Wednesday 26/March 23:59 will not be answered, as the homework requirements cannot be changed further than that.
- You are responsible for following any clarifications or additional requirements added during the QA period.

Runtime Example

Expected Output at stdout:

list of ports:
0 I2C empty
1 SPI empty
2 OneWire empty
3 UART empty
Device added.
list of ports:
0 I2C empty
1 SPI empty
2 OneWire occupied DHT11 TempSensor Sensor 0 OFF
3 UART empty
list of Sensors:
DHT11 0 2 OneWire
DHT11: Turning ON.
DHT11 TempSensor Sensor: Temp: 0.92C.
DHT11: Turning OFF.
Device removed.
Exitting ...

Configuration File:

Port Configuration: I2C,SPI,OneWire,UART
# of sensors:1
# of displays:1
# of wireless adapters:1
# of motor drivers:1

Commands:

list ports
addDev DHT11 2 0
list ports
list Sensor
readSensor 0
turnON 2
readSensor 0
rmDev 2
turnOFF 2
rmDev 2
exit

OneWire_2.log

Writing “turnOFF”.
Reading.
Writing “turnON”.
Port Opened.

Other port logs

Port Opened.

Penalties:

Cheating	You get NA from the course
Code is not compilable	-100 pts (You get 0 directly)
Not attending demo / not providing acceptable explanations during demo	-100 pts (You get 0 directly)
Invalid Input/Output format	Fails the whole test
Not a proper OOP design as described above	-50 pts
Too much reliance on AI	-50 pts
Additional library usage (other than the ones described above)	-50 pts
High time or space complexity, or improper usage of data structures	-30 pts
Bad documentation or project format	-20 pts (for each)
Methods in wrong places	-20 pts

PS: Additional grading criteria may apply.